#11
09/11/03

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	Harlan SEXTON et al.		
Application No.:	09/512,620	Group Art Unit:	2127
Filed:	February 25, 2000	Examiner:	Vo, L.
Attorney Docket:	50277-0403		
Client Docket:	OID-1997-048-19		

For: USING A VIRTUAL MACHINE INSTANCE AS THE BASIC UNIT OF USER EXECUTION IN A SERVER ENVIRONMENT

APPEAL BRIEF

Honorable Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

This Appeal Brief is submitted, in triplicate, in support of the Notice of Appeal filed July 7, 2003.

I. REAL PARTY IN INTEREST

Oracle International Corp. is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals and interferences.

09/09/2003 AWONDAF1 00000013 09512620
01 FC:1402 320.00 OP

III. STATUS OF THE CLAIMS

Claims 1-16 are pending in this appeal. No claim is allowed. This appeal is therefore taken from the final rejection of claims 1-16 on April 3, 2003.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been filed after final rejection.

V. SUMMARY OF THE INVENTION

The present invention addresses problems associated with implementing a virtual machine, especially a JAVA virtual machine that is responsible for executing JAVA problems. (Spec., p. 4) In particular, the present invention addresses problems in conventional virtual machines by providing a scalable server that avoids the performance penalties associated with threads competing for resources. (Spec., p. 7)

To avoid such performance penalties, the Specification discloses an embodiment in which "the overhead associated with each VM instance is reduced by sharing certain data with other VM instances. The memory structure that contains the shared data is referred to herein as the shared state area. Each VM instance has read-only access to the data that has been loaded into the shared state area, and therefore the VM instances do not contend with each other for access rights to that data." (Spec., p. 15) For example, as shown in FIG. 3, there are clients that have established three sessions through a server, in which session 1 has a call being processed by VM instance 1 and session 3 has a call being processed using VM instance 3. "Both VM instance 1 and VM instance 3 share access to the shared state area, which in the illustrated embodiment includes data for Java class X." (Spec., p. 16)

This solution is recited in independent claims 1 and 9 as follows: "wherein said first virtual machine instance and said second virtual machine instance are two of a plurality of virtual machine instances, associated with said server, that share access to data stored in a shared state area allocated in volatile memory associated with said server."

VI. ISSUES

Whether claims 1 and 9 are obvious under 35 U.S.C. § 103 based on *Bayeh* (US 6,223,202) in view of *Johnson et al.* (US 6,330,709)?

Whether claims 2-5 and 10-13 are obvious under 35 U.S.C. § 103 based on *Bayeh* and *Johnson et al.* in view of *Bugnion et al.* (US 6,975,938)?

Whether claims 6 and 14 are obvious under 35 U.S.C. § 103 based on *Bayeh* and *Johnson et al.* in view of *Miner et al.* (US 6,047,053)?

Whether claims 7 and 15 are obvious under 35 U.S.C. § 103 based on *Bayeh* and *Johnson et al.* in view of *Loomans* (US 6,393,605)?

Whether claims 8 and 16 are obvious under 35 U.S.C. § 103 based on *Bayeh* and *Johnson et al.* in view of *Heiney et al.* (US 6,401,109)?

VII. GROUPING OF CLAIMS

The claims should not be regarded as all standing together since the claims recite respective limitations that render each of the claims separately patentable. For the purposes of this appeal, the following groups are recognized:

A. Claims 1 and 9

B. Claims 2-5 and 10-13

C. Claims 6 and 14

D. Claims 7 and 15

E. Claims 8 and 16

VIII. ARGUMENT

A. CLAIMS 1-16 ARE NOT OBVIOUS BECAUSE BOTH *BAYEH* TEACHES AGAINST TWO VIRTUAL MACHINE INSTANCES "THAT SHARE ACCESS TO DATA STORED IN A SHARED STATE AREA ALLOCATED IN VOLATILE MEMORY ASSOCIATED WITH SAID SERVER."

The initial burden of establishing a *prima facie* basis to deny patentability to a claimed invention under any statutory provision always rests upon the Examiner. *In re Mayne*, 41 USPQ2d 1451 (Fed .Cir. 1997); *In re Deuel*, 34 USPQ2d 1210 (Fed. Cir. 1995); *In re Bell*, 26 USPQ2d 1529 (Fed. Cir. 1993); *In re Oetiker*, 24 USPQ2d 1443 (Fed. Cir. 1992). In rejecting a claim under 35 U.S.C. § 103, the Examiner is required to provide a factual basis to support the obviousness conclusion. *In re Warner*, 154 USPQ 173 (CCPA 1967); *In re Lunsford*, 148 USPQ 721 (CCPA 1966); *In re Freed*, 165 USPQ 570 (CCPA 1970). The Examiner is required to show that all the claim limitations are taught or suggested by the references. *In re Royka*, 180 USPQ 580 (CCPA 1974); *In re Wilson*, 165 USPQ 494 (CCPA 1970).

Obviousness rejections require some evidence in the prior art of a teaching, motivation, or suggestion to combine and modify the prior art references. See, e.g., *McGinley v. Franklin Sports, Inc.*, 262 F.3d 1339, 1351-52, 60 USPQ2d 1001, 1008 (Fed. Cir. 2001); *Brown & Williamson Tobacco Corp. v. Philip Morris Inc.*, 229 F.3d 1120, 1124-25, 56 USPQ2d 1456, 1459 (Fed. Cir. 2000); *In re Dembiczak*, 175 F.3d 994, 999, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999). It is improper to combine references where the references teach away from their combination. *In re Grasselli*, 713 F.2d 731, 218 USPQ 769 (Fed. Cir. 1983). A reference must be considered in its entirety, including those portions that would lead away from the claimed

invention. *W.L. Gore & Assocs., Inc. v. Garlock, Inc.*, 721 F.2d 1540, 220 USPQ 303 (Fed. Cir. 1983), *cert. denied*, 469 U.S. 851 (1984).

The rejection of claims under §103 should be reversed because *Bayeh* teaches against the invention recited in the claims. For example, independent claim 1 recites:

> 1. A method for servicing requests received by a server in a multiple-user environment, the method comprising the steps of:
> establishing a first session between said server and a first user;
> establishing a second session between said server and a second user;
> responding to requests that are received by said server in said first session by executing virtual machine code using a first virtual machine instance; and
> responding to requests that are received by said server in said second session by executing virtual machine code using a second virtual machine instance;
> wherein said first virtual machine instance and said second virtual machine instance are distinct instances of a same type of virtual machine;
> wherein said first virtual machine instance exists within said server concurrently with said second virtual machine instance; and
> wherein said first virtual machine instance and said second virtual machine instance are two of a plurality of virtual machine instances, associated with said server, that **share access to data stored in a shared state area** allocated in volatile memory associated with said server.

Independent claim 9, a computer-readable medium claim that corresponds to claim 1, also recites "two of a plurality of virtual machine instances ... that **share access to data stored in a shared state area** allocated in volatile memory." The Examiner correctly acknowledged that *Bayeh* does not teach this feature: "Bayeh ... fails to explicitly teaches of [*sic*] wherein said first machine instance and said second virtual machine instance are two of a plurality of virtual machine instances, associated with said server that share access to data stored in a shared state area allocated with said server." (Office Action of April 3, 2003, p. 3).

In fact, *Bayeh* teaches against this limitation. *Bayeh* describes a technique of virtual machine pooling "for enabling multiple virtual machines to execute on a single server." (Abstract) To avoid problems with one servlet from being able to "overwrite any location in the

memory available to the process, even if another servlet depends on the contents of that memory location being unchanged" (col. 3:18-21), *Bayeh* describes a "protection mechanism to be implemented that **prohibits** the servlets of one application from interfering with the servlets of another application." (ll. 47-50, emphasis added). In fact, this protection mechanism is an object of *Bayeh*: "It is another **object** of the present invention to provide these multiple virtual machines in a way that **protects the integrity of application data** such that applications do overwrite each other's data." (ll. 3:64-67) To address the object of integrity protection of the *Bayeh* system, *Bayeh* declares that this is accomplished by preventing the sharing of memory:

> Further, pooling the virtual machines solves the problem of threads being able to inadvertently overwrite each other's data. Because the threads of one application can now run in a separate virtual machine from the threads of another application, and an application has access only to the resources of its own virtual machine, the resources of the applications are protected from other applications automatically. **No longer can one application write into the memory being used by another application**, because different memory locations are assigned to each virtual machine. One application's open files are no longer available to other applications, because the file is seen as being open only within the VM that opened it. (col. 11:22-35, emphasis added)

Accordingly, *Bayeh* explicitly teaches against the invention recited in independent claims 1 and 9 with the following limitation: "wherein said first virtual machine instance and said second virtual machine instance are two of a plurality of virtual machine instances, associated with said server, that **share access to data stored in a shared state area** allocated in volatile memory associated with said server." Because of this teaching against the claimed invention, there is no motivation for one of ordinary skill in the art to combine *Bayeh* with *Johnson et al.* in order to share access to data, as the Examiner contends, even if the Examiner be correct about the alleged teachings and advantages of *Johnson et al.* (see *infra*).

Actually, the Examiner's motivation for modifying *Bayeh* in view of *Johnson et al.* is confused. First, the Examiner states that "Johnson et al. teaches of [*sic*] accessing **data** stored in

shared address space" (Office Action, p. 3, emphasis added), but then contends that it would have obvious to combine *Bayeh* and *Johnson et al.* "to share access to common **code** stored in state area" (*id.*, emphasis added). This reasoning employs a *non sequitur*, switching from "data" to "code," and is therefore invalid. Perhaps is the Examiner is confused about the difference between data and code; or perhaps the Examiner is well aware of both the difference between code and data and *Bayeh*'s teaching against sharing data and so switched to talking about code, but regardless of the cause of the *non sequitur*, independent claims 1 and 9 recite "share access to data" and *Bayeh* et al. teaches against sharing data.

B. CLAIMS 1-16 ARE ALSO NOT OBVIOUS BECAUSE *JOHNSON ET AL.* DOES NOT TEACH OR SUGGEST TWO VIRTUAL MACHINE INSTANCES "THAT SHARE ACCESS TO DATA STORED IN A SHARED STATE AREA ALLOCATED IN VOLATILE MEMORY ASSOCIATED WITH SAID SERVER."

As argued above, *Bayeh* does not teach, and in fact teaches against, "two of a plurality of virtual machine instances ... that share access to data stored in a shared state area allocated in volatile memory." This too is not found in *Johnson et al.*

Johnson et al. is directed to a "virtual machine for shared persistent objects" (Title) and describes the design of "several objects used to implement persistent objects in a shared address space" (Abstract; col. 3:41-43). The importance of persistent objects to the *Johnson et al.* system is stated on col. 2:1-14 as follows, with added emphasis:

> One issue in object oriented programming, and Java programing [*sic*] in particular, is object persistence. Typically, when a process ends, all of the references to objects created by the process are lost, and the objects themselves are destroyed. These objects are referred to as transient objects. Persistent objects, as opposed to transient objects, have a lifetime that transcends the lifetime of the process that created them. To make an object persistent, **mechanisms must be put in place** to allow a persistent object to survive beyond the lifetime of the process from which the object was created so that other processes can access the

> object. This typically involves the storing of the objects onto **permanent storage devices**, such as hard disks, optical disks, tape drives, etc.

To this end, *Johnson et al.* discusses the use of a shared address space (SAS) **220** that is responsible for performing address translations and controls the retrieval of data from a permanent data store **206** (col. 11:2-5) and defines its term "shared address space" as: "in this application the term **shared address space** refers to the large address space that allows applications to store **persistent** data using single level store semantics" (col. 12:1-4, emphasis added).

As to how this shared address space (SAS) can be used with a Java Virtual Machine (JVM), *Johnson et al.* discloses in a passage cited by the Examiner that the "JVM can store objects in **either** temporary local storage **or** in **permanent** SAS storage **220**" (col. 20:9-10). Neither memory satisfy the claim limitation "a shared state area allocated in volatile memory." The temporary local storage is not "shared", and the SAS storage **220** is persistent, not "in volatile memory.' Therefore, *Johnson et al.* fails to teach "two of a plurality of virtual machine instances ... that share access to data stored in a shared state area allocated in volatile memory."

Furthermore, the use of the shared address space (SAS) is only disclosed in *Johnson et al.* to be between a JVM and an application (col. 20:7-20), not between a "**plurality of virtual machine instances**" as recited in the claims. The portion of *Johnson et al.* cited by the Examiner (col. 6:15-19) fails to indicate otherwise: "operations in Java are performed by one object calling a method on another object. These objects can reside locally on the same machine or on separate JVMs physically located on separate computers or system." (col. 6:15-19). Contrary to the Examiner's conjecture that "this indicates that objects on separate JVMs can also be shared as well," (p. 9), *Johnson et al.* does not disclose any details on how an object can call a method on an object on a separate computer. Presumably, the remote call would be accomplished in a

conventional manner, i.e. with a remote procedure call that transmits a copy of the object not by using "access to data stored in a shared state area in volatile memory." Accordingly, *Johnson et al.* fails to evidence sharing data stored in a volatile memory between two virtual machine instances.

IX. CONCLUSION AND PRAYER FOR RELIEF

Appellants, therefore, request the Honorable Board to reverse each of the Examiner's rejections.

Respectfully Submitted,

DITTHAVONG & CARLSON, P.C.

9/5/2003
Date

Stephen C. Carlson
Attorney for Applicant(s)
Reg. No. 39929

10507 Braddock Rd, Suite A
Fairfax, VA 22032
Tel. 703-425-8516
Fax. 703-425-8518

APPENDIX

1. A method for servicing requests received by a server in a multiple-user environment, the method comprising the steps of:

establishing a first session between said server and a first user;

establishing a second session between said server and a second user;

responding to requests that are received by said server in said first session by executing virtual machine code using a first virtual machine instance; and

responding to requests that are received by said server in said second session by executing virtual machine code using a second virtual machine instance;

wherein said first virtual machine instance and said second virtual machine instance are distinct instances of a same type of virtual machine;

wherein said first virtual machine instance exists within said server concurrently with said second virtual machine instance; and

wherein said first virtual machine instance and said second virtual machine instance are two of a plurality of virtual machine instances, associated with said server, that share access to data stored in a shared state area allocated in volatile memory associated with said server.

2. The method of Claim 1 further comprising the step of sharing, between said first virtual machine instance and said second virtual machine instance, a set of one or more resources within said shared state area.

3. The method of Claim 2 wherein the step of sharing a set of one or more resources includes sharing data associated with an object class.

4. The method of Claim 1 wherein said plurality of virtual machine instances share read-only access to said data stored in said shared state area allocated in volatile memory within said server.

5. The method of Claim 1 wherein:

said shared state area stores data associated with an object class; and

said first virtual machine instance stores, in session-specific memory associated with said first virtual machine instance, a first value for a static variable associated with said object class; and

said second virtual machine instance stores, in session-specific memory associated with said second virtual machine instance, a second value for said static variable associated with said object class.

6. The method of Claim 1 further comprising the steps of:

responding to a call associated with a particular session with said server by allocating a call memory for the particular virtual machine instance associated with said particular session; and

discarding said call memory upon termination of said call.

7. The method of Claim 1 further comprising the step of:

responding to a call associated with a particular session with said server by scheduling, for execution in a system thread, the particular virtual machine instance associated with said particular session.

8. The method of Claim 1 further comprising the steps of:

spawning the first virtual machine instance by instantiating a data structure associated with a single session; and

storing a pointer within said data structure to provide access to the data stored in the shared state area.

9. A computer-readable medium carrying instructions for servicing requests received by a server in a multiple-user environment, the instruction comprising instructions for performing the steps of:

establishing a first session between said server and a first user;

establishing a second session between said server and a second user;

responding to requests that are received by said server in said first session by executing virtual machine code using a first virtual machine instance; and

responding to requests that are received by said server in said second session by executing virtual machine code using a second virtual machine instance;

wherein said first virtual machine instance and said second virtual machine instance are distinct instances of a same type of virtual machine;

wherein said first virtual machine instance exists within said server concurrently with said second virtual machine instance; and

wherein said first virtual machine instance and said second virtual machine instance are two of a plurality of virtual machine instances, associated with said server, that share access to data stored in a shared state area allocated in volatile memory associated with said server.

10. The computer-readable medium of Claim 9 further comprising instructions for performing the step of sharing, between said first virtual machine instance and said second virtual machine instance, a set of one or more resources within said shared state area.

11. The computer-readable medium of Claim 10 wherein the step of sharing a set of one or more resources includes sharing data associated with an object class.

12. The computer-readable medium of Claim 9 wherein said plurality of virtual machine instances share read-only access to said data stored in said shared state area allocated in volatile memory within said server.

13. The computer-readable medium of Claim 9 wherein:

said shared state area stores data associated with an object class; and

said first virtual machine instance stores, in session-specific memory associated with said first virtual machine instance, a first value for a static variable associated with said object class; and

said second virtual machine instance stores, in session-specific memory associated with said second virtual machine instance, a second value for said static variable associated with said object class.

14. The computer-readable medium of Claim 9 further comprising instructions for performing the steps of:

responding to a call associated with a particular session with said server by allocating a call memory for the particular virtual machine instance associated with said particular session; and

discarding said call memory upon termination of said call.

15. The computer-readable medium of Claim 9 further comprising instructions for performing the step of:

responding to a call associated with a particular session with said server by scheduling, for execution in a system thread, the particular virtual machine instance associated with said particular session.

16. The computer-readable medium of Claim 9 further comprising instructions for performing the steps of:

spawning the first virtual machine instance by instantiating a data structure associated with a single session; and

storing a pointer within said data structure to provide access to the data stored in the shared state area.